J. Symbolic Computation (1994) **17**, 409–420

Computing a Set of Generators of Minimal Cardinality in a Solvable Group

ANDREA LUCCHINI and FEDERICO MENEGAZZO

Università di Padova, Dipartimento di matematica pura e applicata

(*Received 13 July 1992*)

In this paper two algorithms are presented which compute a set of generators of minimal cardinality for a finite soluble group given by a polycyclic presentation. The first can be used when a chief series is available. The second algorithm is less simple, but nevertheless efficient, and can be used when it is difficult or too expensive to compute a chief series. The problem of determining the minimal number $d(G)$ of generators when G is a solvable group has been discussed and solved by Gaschütz, and the ideas for these algorithms are essentially suggested by the work of Gaschütz. In the Appendices CAYLEY V.3.7.2 procedures are listed.

The problem of determining the minimal number d(G) of generators for a finite solvable group G has been discussed and completely solved by Gaschütz (1959). The work of Gaschütz also suggests a computational method to find a set of generators of minimal cardinality for a finite solvable group. The main idea is to construct a series with abelian factors and to use the following fundamental remark:

PROPOSITION (Gaschütz (1955)). *Let N be a normal subgroup of G and let* $y_1,\ldots,y_d \in G$ *be such that* $G/N=\langle y_1N,\ldots,y_dN\rangle$. *If G can be generated with d elements then there exist* $u_1,\ldots,u_d \in N$ *such that* $G=\langle y_1u_1,\ldots,y_du_d\rangle$.

This suggests a simple algorithm to determine a set of generators of minimal cardinality for a finite solvable group G, when a chief series $1= G_m \lhd G_{m-1} \lhd \ldots \lhd G_1 \lhd G_0= G$ of G is available. Suppose in fact to have already found a minimal generating set $y_1G_i,\ldots,y_dG_i$ for the factor group G/G_i: the next step is to find a minimal generating set for G/G_{i+1}. There are two possibilities: either $d(G/G_i)=d(G/G_{i+1})$ or not. In the first case there exist $u_1,\ldots,u_d \in G_i$ such that $G/G_{i+1}=\langle y_1u_1G_{i+1},\ldots,y_du_dG_{i+1}\rangle$; in the second, since G_i/G_{i+1} is a minimal normal subgroup of G/G_{i+1}, for any $u \in G_i-G_{i+1}$, $y_1G_{i+1},\ldots,y_dG_{i+1}, uG_{i+1}$ is a generating set of minimal cardinality for G/G_{i+1}. So the iterative step in the algorithm is: given a set $y_1,\ldots y_d$ of

0747–7171/94/050409 + 12 $08.00/0

generators of G modulo G_i of minimal cardinality, try to modify these elements multiplying by elements in G_i to get generators modulo G_{i+1}; if this is not possible, then add to $y_1,\dots y_d$ an element y_{d+1} chosen in G_i-G_{i+1}. To make this algorithm really efficient the following remark is very useful :

LEMMA 1.

Let N be a minimal normal subgroup of a finite solvable group G; let $e_1,\dots,e_m$ be a system of generators of N and $y_1N,\dots,y_dN$ a minimal generating set for G/N. If G can be generated by d elements then either $G=\langle y_1,\dots y_d\rangle$ or there exist i, $1\le i\le d$, and j, $1\le j\le m$, such that $y_1,\dots, y_{i-1}, y_ie_j, y_{i+1},\dots, y_d$ is a generating set for G.

The consequence of this Lemma is that when one tries to modify the elements $y_1,\dots,y_d$ multiplying by the d-tuple $u_1,\dots u_d$ of elements in G_i-G_{i+1}, one can restrict the choice to d-tuples of the form $1,\dots,1,e_j,1,\dots,1$ if $e_1,\dots e_m$ is a set of generators of G_i modulo G_{i+1}.

Lemma 1 will be a straightforward corollary of the following:

LEMMA 2.

Let N be an abelian normal subgroup of G; suppose that $e_1,\dots,e_m$ is a system of generators of N and that $G/N=\langle y_1N,\dots y_dN\rangle$. If $\langle y_1,\dots,y_d\rangle \cap N = 1$, but there exist $u_1,\dots,u_d \in N$ such that $\langle y_1u_1,\dots,y_du_d\rangle \cap N \neq 1$, then there exist i, $1\le i\le d$, and j, $1\le j\le m$, such that $\langle y_1,\dots, y_{i-1}, y_ie_j, y_{i+1},\dots, y_d\rangle \cap N \neq 1$.

PROOF.

Suppose $H\cap N=1$, where $H=\langle y_1,\dots y_d\rangle$. Then H is a complement for N in G and for any $\delta\in Der(H,N)$, $H(\delta)=\{xx^\delta | x\in H\}$ is again a complement for N in G: this gives a bijection between Der(H,N) and the set of all complements of N in G. Our assumption means that exists a map $\varepsilon:\{y_1,\dots y_d\}\to N$ such that $\langle y_1y_1^\varepsilon,\dots,y_dy_d^\varepsilon\rangle\cap N\neq 1$. If $\delta\in Der(H,N)$ define $\bar\delta:\{y_1,\dots y_d\}\to N$ to be the restriction of δ to the set $\{y_1,\dots y_d\}$: then $H(\delta)=\{xx^\delta | x\in H\}$ is a complement for N in G and contains $\langle y_1y_1^{\bar\delta},\dots,y_dy_d^{\bar\delta}\rangle$. Since $N\langle y_1y_1^{\bar\delta},\dots,y_dy_d^{\bar\delta}\rangle=G$, it follows that $H(\delta)=\langle y_1y_1^{\bar\delta},\dots,y_dy_d^{\bar\delta}\rangle$. Conversely, if the map $\varepsilon:\{y_1,\dots y_d\}\to N$ is such that

$\langle y_1y_1^{\varepsilon},\ldots,y_dy_d^{\varepsilon}\rangle \cap N=1$, then $\langle y_1y_1^{\varepsilon},\ldots,y_dy_d^{\varepsilon}\rangle$ is a complement for N in G, hence $\langle y_1y_1^{\varepsilon},\ldots,y_dy_d^{\varepsilon}\rangle = H(\delta)$ for some $\delta\in Der(H,N)$: i.e. $y_iy_i^{\varepsilon}=xx^{\delta}$ for some $x\in H$, $1\le i\le d$, so $x^{-1}y_i=x^{\delta}(y_i^{\varepsilon})^{-1}\in H\cap N$, hence $x=y_i$, $y_i^{\varepsilon}=y_i^{\delta}$ and $\varepsilon=\bar{\delta}$. Therefore:$\langle y_1y_1^{\varepsilon},\ldots,y_dy_d^{\varepsilon}\rangle\cap N=1$ if $\varepsilon=\bar{\delta}$ for some $\delta\in Der(H,N)$, $\langle y_1y_1^{\varepsilon},\ldots,y_dy_d^{\varepsilon}\rangle\cap N\neq 1$ if $\varepsilon\notin\{\bar{\delta}|\delta\in Der(H,N)\}$. The map from Der(H,N) into $N^{\{y_1,\ldots,y_d\}}$ which sends δ to $\bar{\delta}$ is a homomorphism of abelian groups and we know that this homomorphism is not surjective; on the other hand, if $e_1,\ldots,e_m$ is a set of generators for N, then $N^{\{y_1,\ldots,y_d\}}$ is generated by the maps $f_{i,j}$, $1\le i\le d$, $1\le j\le m$, defined by $f_{i,j}(y_r)=\delta_{i,r}e_j$. Therefore there exist i, j such that $f_{i,j}\notin \overline{Der(H,N)}$, which implies $\langle y_1y_1^{f_{i,j}},\ldots,y_dy_d^{f_{i,j}}\rangle\cap N\neq 1$. #

In Appendix A there is the listing of a procedure called MINGEN, written in CAYLEY V3.7.2, which implements this algorithm: it requires as input a polycyclic presentation of a finite solvable group.

This first algorithm requires the computation of the chief series of the group G. The same idea can be modified to produce another algorithm that computes a set of generators of minimal cardinality using only a series with abelian factors (for example the derived series). In this case the task for the iterative step is: given an abelian normal subgroup N of a finite group G and a set $Y=\{y_1,\ldots,y_d\}$ of minimal cardinality with respect to the property $G=\langle y_1,\ldots,y_d,N\rangle$, find a minimal set of generators Y* for the group G.

Now in general N is not a minimal normal subgroup, so it is no longer possible to modify the set Y into a set of generators for G in one only step: but the idea is to develop an algorithm that, given Y and N, produces Y*, N* such that N* is a normal subgroup of G, with $N^*\lneq N$, and Y* is a minimal generating set for G modulo N*: one then repeats this algorithm until N*=1.

Given the (non trivial) abelian normal subgroup N of G and Y, a generating set for G modulo N of minimal cardinality, we construct two sequences:

$$1= N_0,N_1,N_2,\ldots,N_m \qquad\qquad Y= Y_1,Y_2,\ldots,Y_m$$

with the properties:

i) $1=N_0\le N_1\lneq N_2\lneq \ldots \lneq N_m=N$ is a series of normal subgroups of G with $N_i\neq N_{i-1}$ if $i\ge 2$;

ii) $|Y_i| \leq d(G/M)$ for every normal subgroup M of G with $N_{i-1} \leq M \not\geq N$;

iii) $\langle Y_i, N \rangle = G$;

iv) $N_i = \langle Y_i \rangle N_{i-1} \cap N$.

Define $Y^* = Y_m$ and $N^* = N_{m-1}$: by (iii) $G = \langle Y^*, N \rangle$ and by (iv) $N \leq \langle Y^* \rangle N^*$ so $G = \langle Y^*, N^* \rangle$; on the other hand, by (ii), $|Y^*| \leq d(G/N^*)$, therefore Y^* is a minimal generating set for G modulo N^*.

We now describe the procedure to construct the two sequences: to begin we set $N_0 = 1$, $Y_1 = Y$, $N_1 = \langle Y \rangle \cap N$. The iterative step that constructs Y_{i+1} (and N_{i+1}) from Y_i, N_i and N_{i-1} is the following: N_i is a normal subgroup of G and $\overline{H} = \langle Y_i \rangle N_i / N_i$ is a complement for the abelian normal subgroup $\overline{N} = N/N_i$ in $\overline{G} = G/N_i$. Suppose $Y_i = \langle y_1, \ldots, y_d \rangle$ and consider the d-tuples $\overline{y_1}\,\overline{u_1}, \ldots, \overline{y_d}\,\overline{u_d}$ ($\overline{y_k} = y_k N_i$, $\overline{u_k} = u_k N_i \in \overline{N}$). There are two possibilities:

a) for at least one choice of $\overline{u_1}, \ldots, \overline{u_d}$, $\langle y_1 u_1, \ldots, y_d u_d, N_i \rangle \cap N \not\geq N_i$: in this case take $Y_{i+1} = \{y_1 u_1, \ldots, y_d u_d\}$; it is clear that $\langle Y_{i+1}, N \rangle = \langle Y_i, N \rangle = G$; furthermore, since by construction, $|Y_i| \leq d(G/M)$ for every normal subgroup M of G with $N_{i-1} \leq M \not\geq N$, and $N_{i-1} \leq N_i$, we get also $|Y_{i+1}| = |Y_i| \leq d(G/K)$ for every normal subgroup K of G with $N_i \leq K \not\geq N$, so (ii) is verified.

b) each d-tuple $y_1 u_1, \ldots, y_d u_d$ generates again a complement for N in G modulo N_i: in this case from the Proposition we can conclude that, for every normal subgroup M of G with $N_i \leq M \not\geq N$, the quotient G/M cannot be generated with d elements; we take $Y_{i+1} = \langle y_1, \ldots, y_d, y_{d+1} \rangle$, where y_{d+1} is an arbitrary element in $N - N_i$. Obviously $\langle Y_{i+1}, N \rangle = G$ and, by the previous remark, $d(G/M) \geq d+1 = |Y_{i+1}|$ for every normal subgroup M of G with $N_i \leq M \not\geq N$.

Then, in both cases, set $N_{i+1} = \langle Y_{i+1} \rangle N_i \cap N$; note that if $N_i \not\geq N$ then $N_i \not\geq N_{i+1}$.

To implement this algorithm Lemma 2 is again useful: if $\overline{e_1}, \ldots, \overline{e_m}$ is a set of generators for the abelian group $\overline{N}$, then to check whether there exists or not a d-tuple $\overline{u_1}, \ldots, \overline{u_d}$ such that $\langle \overline{y_1}\,\overline{u_1}, \ldots, \overline{y_d}\,\overline{u_d} \rangle$ is not a complement for $\overline{N}$ it is enough to consider only those d-tuples $\overline{u_1}, \ldots, \overline{u_d}$ of the form $\overline{u_k} = 1$, $k \neq j$, $\overline{u_j} = \overline{e_i}$, $1 \leq j \leq d$, $1 \leq i \leq m$.

We have described above the process to construct N^*: we have seen that it produces not

only N* and Y*, but also a series $N_0 \leq N_1 \leq \ldots \leq N_{m-1}=N^*$ of normal subgroups of G contained in N*. Our algorithm is organized in such a way to "remember" these subgroups when passing from N* to (N*)*: to do so the main idea is to work modulo N_{m-2} in the step which constructs (N*)* (see, in Appendix B, the listing of the procedure GEN1 for more details).

In Appendix B are shown the listings of some procedures, written in CAYLEY V.3.7.2, that implement this algorithm. They all work for solvable groups given by polycyclic presentations. The procedure GEN(G,M,mingen;mingen) needs as input a polycyclic presentation for a solvable group G, a normal subgroup M of G and a set "mingen" of elements generating G modulo M of minimal cardinality; it works along the derived series of M, calling GEN1 to produce generators of $G/M^{(i+1)}$ given those of $G/M^{(i)}$. This can be used if for some reason a minimal set of generators for some quotient G/M is already known or easier to compute with different strategies (this happens for example if G/M is nilpotent). Otherwise the procedure GENER(G;setgen) computes a set "setgen" of generators for G modulo the derived subgroup G' (using the subroutine ABGEN) and then calls GEN(G,G',setgen;mingen).

Denote with $\Phi(G)$ the Frattini subgroup of the group G. As it is well known if $g_1,\ldots,g_n$ is a set of generators of G modulo a normal subgroup N and N is contained in $\Phi(G)$, then $g_1,\ldots,g_n$ is also a set of generators for G; so, whenever a normal subgroup N contained in $\Phi(G)$ is known, our algorithm can be applied to the quotient G/N instead than to G. In particular a possibility is to first compute $\Phi(G)$ (a practical algorithm exists for doing that (Glasby (1988)) and then find a minimal generating set for $G/\Phi(G)$. This strategy is certainly the best when G is a p-group (or more generally a nilpotent group); in fact in this case not only the Frattini subgroup $\Phi(G)$ is easier to compute, but also $G/\Phi(G)$ is an abelian group (so for example generators for $G/\Phi(G)$ can be computed using the procedure ABGEN).

The following table describes the performances of our algorithms for some running samples (using Cayley V.3.7.2 on a VAX-VS 3100/76). Some of the examples are wreath product HwrK of permutation groups, where H and K are chosen to be either S_n, the symmetric group of degree n, D_n, the dihedral group of order 2n, thought as a permutation

group of degree n, or C_n, the cyclic group of order n. Moreover the direct products P^n, $1 \leq n \leq 6$, are considered of n copies of the group P constructed extending with an involution the direct product of the non abelian group of order 27 and exponent 3 with a cyclic group of order 3 .

For each of these groups, in the first column of the table we record the minimal number of generators (in the case of P^n, one gets $d(P^n)=4$ for $n \leq 3$, $d(P^n)=n$ if $n \geq 4$: this agrees with a result of Wiegold (1974): if $G \neq G'$ then there exists k such that $d(G^r)=r \cdot d(G/G')$ for any $r \geq k$). The second and the forth columns contain the timings t_1, t_2, expressed in CPU seconds, spent to compute a minimal set of generators using the MINGEN algorithm or the GENER algorithm rispectively.

A consistent fraction of t_1 is needed to compute the chief series: this timing s_1 is written in the third column of the table. The efficency of the algorithm MINGEN strongly depends on this time s_1: when the computation of the chief series becomes too long, as for example in the case of $S_4 wr D_{12}$, the GENER algorithm completes faster than MINGEN.

G	d(G)	t_1	s_1	t_2
$S_4 wr S_4$	2	16	6.8	18
$S_4 wr D_{12}$	3	302	125	197
$C_{21} wr C_{14}$	2	171	93	85
$C_{14} wr C_7$	2	15	7	3.7
P	4	1.4	0.5	3.1
p2	4	5.5	1.6	9.8
p3	4	15	4.1	21
p4	4	34	9	48
p5	5	69	18	93
p6	6	128	31	165

A version of the program GENER has been implemented also in the computational system GAP 3.2 (we have not implemented MINGEN since a command to compute the chief series is not yet available in GAP). The performances on some running samples are very satisfactory ; for example a minimal number of generators for the wreath product $S_4 wr C_{21}$ (which has order $2^{63}3^{22}7$) can be computed in only 56 CPU seconds (unfortunately, we could only use CAYLEY on a VAX, and GAP on a SUN OS; so we are not in condition to compare the performances of the two systems).

References

Gaschütz, W. (1955). Zu einem von B. H. Neumann und H. Neumann gestellten Problem. *Math. Nachr.* **14**, 249-252

Gaschütz, W. (1959). Die Eulersche Funktion endlicher auflösbarer Gruppen. *Illinois J. Math.* 3, 469-476

Wiegold, J. (1974). Growth sequences of finite groups II. *J. Australian Math. Soc.* **20** 225-229

Glasby, S. (1988) Computational approaches to finite group theory, University of Sydney.

Appendix A

```
Procedure MINGEN(G;gen);

Given a soluble group G, it returns a minimal set "gen" of
generators of G.

Distinguish the easy case when G is a p-group
pr=seqset(pcprimes(G));
if card(pr) eq 1
    then F=frattini(G);
    Q,h=G/F;
    pcGen=pcgenerators(Q);
    gen=h@(pcGen);
    print gen;
    return;
end;

L=chief series(G);
pcGen=pcgenerators(G);
```

```
Find a minimal generating set for G mod L[2]; since G/L[2] is cycl
of prime order it is enough to find an element of the set pcGen no
in L[2]

for each x in pcGen do
    if x in L[2] then loop;
    else
        gen=[x]; break;
    end;
end;

for i=3 to length(L) do
    Q,f=G/L[i];
    j=i-1;
    N=f(L[j]);
    Qgen=f(gen);
    H=<Qgen>;
    if H eq Q
        then Qggen=Qgen;
        goto 1;
    end;
    Qgen is a minimal set of generators for Q mod N; construct
    Qggen, a minimal set of generators for Q;
    theSequence=setseq(Qgen);
    Ngen=pcgenerators(N);
    for each r in Ngen do
        for s=1 to length(theSequence)  do
            y=r*theSequence[s];
            otherSequence=theSequence;
            otherSequence[s]=y;
            otherGen=seqset(otherSequence);
            H=<otherGen>;
            if H eq Q then
                Qggen=otherGen;
                goto 1;
            end;
        end;
    end;
    theSequence=append(theSequence,Ngen[1]);
    Qggen=seqset(theSequence);
1:  gen=f@(Qggen);
    end;
print gen;
end;"procedure MINGEN";
```

Appendix B

```
procedure GENER(G;setgen);

Given a soluble group, it returns a minimal set of generators for G

Distinguish the easy case when G is a p-group
pr=seqset(pcprimes(G));
if card(pr) eq 1 then
    F=frattini(G);
    Q,ff=G/F;
    gen=pcgenerators(Q);
    setgen=ff@(gen);
    return;
end;

M=derived subgroup(G);
Q,f=G/M;
ABGEN(Q;gen);
mingen=f@(gen);
mingen is a minimal set of generators of G modulo M; call the
routine GEN to construct from it a minimal set of generators of G
GEN(G,M,mingen;setgen);
print setgen;
end; "procedure GENER"


procedure GEN(G,M,mingen;mingen);

The input data should satisfy the following conditions:
G is a soluble group,
M is a normal subgroup of G,
mingen is a minimal set of generators of G mod M (not quite, SOME
redundancy is allowed as mingen=[identity] when G=M).
The procedure returns a minimal set of generators of G

L=M;
while L ne <identity of G> do
    N=derived subgroup(L);
    GEN1(G,L,N,mingen; mingen);
    L=N;
end;
end; "procedure GEN"
```

```
procedure GEN1(G,L,M,gen;gen);
The input data must satisfy the following conditions:
G is a soluble group,
L>M are normal subgroups of G with L/M abelian;
gen is a minimal set of generators of G modulo L.
The procedure returns a minimal set of generators of G modulo M

Work on the quotient G/M

QQ,ff=G/M;
ABFRAT(ff(L);N);

It sufficies to find a generating set for QQ modulo N

Q,f=QQ/N;
LL=f(ff(L));
NN=f(N);
qgen=f(ff(gen));
norSeq=seq(NN);
Y=LL;
while Q ne <qgen> do
    K=norSeq[length(norSeq)];
    KK=<qgen, K> meet Y;
        if KK ne K then
        norSeq=append(norSeq,KK);
    end;
    while KK ne Y do
        K=norSeq[length(norSeq)];
        SUPPL(Q,Y,K,qgen;qgen);
        KK=<qgen,K> meet Y;
        norSeq=append(norSeq,KK);
    end;
    norSeq=prune(prune(norSeq));
    Y=K;
end;
gen=ff@(f@(qgen));
end;"procedure GEN1"
```

```
procedure SUPPL(G,M,N,complGen;gen);

The following conditions are assumed on the input data:
G is a soluble group,
M>N are normal subgroups of G
the factor group M/N is abelian,
complGen is a set of elements of G that generate a complement of
M/N in G/N.
The function returns  a list of elements of G that generate a
supplement of M/N in G/N that is not a complement
Q,f=G/N;
MM=f(M);
gen=f(complGen);
Abgen(MM;genMM);
theSequence=setseq(gen);
for each r in genMM do
    for s=1 to length(theSequence) do
        y=r*theSequence[s];
        otherSequence=theSequence;
        otherSequence[s]=y;
        otherGen=seqset(otherSequence);
        HH=<otherGen> meet MM;
        if order(HH) ne 1 then
            gen=f@(otherGen);
            return;
        end;
    end;
end;
theSequence=append(theSequence, genMM[1]);
gen=f@(seqset(theSequence));
end;"procedure SUPPL"
```

```
procedure ABGEN(G;abgen);
G must be an abelian group; the procedure returns a minimal
set abgen of generators for G

We compute first a sequence gen consisting of the elements of
a minimal generating set for the quotient G/FratG
ABFRAT(G;N);
Q,f=G/N;
pr=pcprimes(Q);
prSet=seqset(pr);
gen=empty;
compute the exponent e of Q
e=1;
for each p in prSet do
    e=e*p;
end;
theSequence=conseq(0,length(pr));
while pr ne theSequence do
    x=identity of Q;
    for each p in prSet do
        i=position(pr,p);
        if i ne 0
            then m=e/p;
            x=x*(Q.i)^m;
            pr[i]=0;
        end;
    end;
    gen=append(gen,x);
end;
abgen=f@(seqset(gen));
end;"procedure ABGEN"

procedure ABFRAT(G;N);

G is a finite abelian group; the procedure returns N=Frat(G)

prSet=seqset(pcprimes(G));
s=1;
for each p in prSet do
    s=s*p;
end;
gset=(pcgenerators(G))^s;
N=<gset>;
end;"procedure ABFRAT"
```